FILE

IN

C

C++

VC++

نام محقق : محمد بشیری

آدرس پست الکترونیک: m.bashiry@gmail.com

آدرس سایت: http://bashiry.250free.com

تاریخ تالیف: ۸۴/۱۰/۹

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

فایل در زبان های برنامه نویسی C، C++ و VC++

هر فایل شامل مجموعه ای از داده های مرتبط به هم است، مانند داده های مربوط به کلیه دانشجویان یک دانشگاه. فایل یک نوعی از ساختمان داده است که این نوع از ساختمان داده، بر روی حافظه جانبی مثل دیسک نوار و غیره تشکیل می شود. چون اطلاعات موجود در روی حافظه جانبی با قطع جریان برق قطع اجرای برنامه و یا دلایلی از این قبیل از بین نمی روند به دفعات زیادی مورد استفاده قرار می گیرند.

در زبان برنامه نویسی C فایل داده می تواند هر دستگاهی مثل صفحه نمایش، صفحه کلید، چاپگر، ترمینال، دیسک، نوار و غیره باشد. داده ها ممکن است به چهار روش در فایل ذخیره شده سپس بازیابی شوند:

۱- داده ها، کاراکتر به کاراکتر در فایل نوشته شده سپس کاراکتر به کاراکتر از فایل خوانده شوند.
۲- داده ها به صورت رشته ای از کاراکترها در فایل نوشته شده سپس به صورت رشته ای از کاراکترها دستیابی شوند.
۳- داده ها در حین نوشتن بر روی فایل، با فرمت خاصی نوشته شده سپس با همان فرمت خوانده شوند (کاراکتری، عددی، رشته ای).
۴- داده ها به شکل ساختمان (رکورد) بر روی فایل نوشته شده سپس بصورت ساختمان از فایل خوانده شوند.

انواع فایل از نظر نوع اطلاعات

داده ها ممکن است در فایل به دو صورت ذخیره شوند:

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

۱- اسکی[1] یا متن

۲- باینری[2]

این دو روش ذخیره شدن داده ها، در موارد زیر با یکدگر تفاوت دارند:

۱- تعیین انتهای خط

۲- تعیین انتهای فایل

۳- نحوه ذخیره شدن اعداد بر روی دیسک

در فایل متنی، اعداد به صورت رشته ای از کاراکترها ذخیره می شوند ولی در فایل باینری اعداد به همان صورتی که در حافظه قرار می گیرند بر روی دیسک ذخیره می گردند.

به عنوان مثال، در فایل متنی عدد ۲۵۶ سه بایت را اشغال می کند. زیرا هر رقم آن، به صورت یک کاراکتر در نظر گرفته می شود. ولی در فایل باینری این عدد در دو بایت ذخیره می شود. (چون عدد ۲۵۶ یک عدد صحیح است و اعداد صحیح در دو بایت ذخیره می شوند).

در فایل متنی کاراکتری که پایان خط را مشخص می کند، در حین ذخیره شدن بر روی دیسک، باید به کاراکترهای CR/LF (Carriage Return Line Feed) تبدیل شود و در حین خوانده شدن، عکس این عمل باید صورت گیرد: یعنی کاراکترهای CR/LF باید به کاراکترهای تعیین کننده پایان خط تبدیل شوند و بدیهی است که این تبدیلات مستلزم صرف وقت است، لذا دسترسی به اطلاعات موجود در فایلهای متنی کندتر از فایل های باینری است.

اختلاف دیگر در فایلهای متنی و باینری در **تشخیص انتهای فایل** است. در هر دو روش ذخیره فایلها، طول فایل توسط سیستم نگهداری می شود و انتهای فایل با توجه به طول فایل مشخص می گردد. در حالت متنی کاراکتر 1A (در مبنای ۱۶) و یا ۲۶ (در مبنای ۱۰) مشخص کننده انتهای فایل است. (این کاراکتر با فشار دادن کلید CTRL به همراه Z ایجاد می شود).

---

[1] ASCII
[2] Binary

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

حال به بررسی توابع و دستورات مختلف فایل در زبان C یا C++ می پردازیم:

باز کردن فایل

هر فایل قبل از اینکه بتواند مورد استفاده قرار گیرد باید باز شود، مواردی که حین بازکردن فایل مشخص می شود عبارتند از:

۱- نام فایل

۲- نوع فایل از نظر ذخیره اطلاعات متنی یا باینری

۳- نوع فایل از نظر ورودی-خروجی (آیا فایل فقط به عنوان ورودی است، آیا فقط به عنوان خروجی است یا هم به عنوان ورودیست و هم به عنوان خروجی است).

یک فایل ممکن است طوری باز شود که فقط عمل نوشتن اطلاعات بر روی آن مجاز باشد. به چنین فایلی فایل خروجی گفته می شود. اگر فایل طوری باز شود که فقط عمل خواندن اطلاعات از آن امکان پذیر باشد به چنین فایلی، فایل ورودی گفته می شود. اگر فایل طوری باز شود که هم عمل نوشتن اطلاعات بر روی آن مجاز باشد و هم عمل خواندن اطلاعات از آن،به چنین فایلی، ورودی-خروجی گفته می شود.

برای باز کردن فایل از تابع fopen() استفاده می گردد. این تابع در فایل stdio.h قرار دارد و به صورت زیر به کار می رود.

FILE *fopen (char *filename, *mode)

در این الگو، filename به رشته ای اشاره می کند که حاوی نام فایل و محل تشکیل یا وجود آن است.

Mode مشخص می کند که فایل چگونه باز شود (ورودی، خروجی و یا ورودی-خروجی).

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

مقادیری که می توان به جای عبارت mode قرار داد در جدول زیر به طور کامل آورده شده است:

MODE	مفهوم
r(rt)	فایلی از نوع text را به عنوان ورودی باز می کند
w(wt)	فایلی از نوع text را به عنوان خروجی باز می کنند
a(at)	فایل را طوری باز می کند که بتوان اطلاعاتی را به آن اضافه کرد.
rb	فایلی از نوع باینری را به عنوان ورودی باز می کند.
wb	فایلی از نوع باینری را به عنوان خروجی باز می کند.
ab	فایل موجود از نوع باینری را طوری باز می کند که بتوان اطلاعات را به انتهای آن اضافه نمود.
r+(r+t)	فایل موجود از نوع text را به عنوان ورودی و خروجی باز می کند
w+(w+t)	فایلی از نوع text را به عنوان ورودی و خروجی باز می کند
a+(a+t)	فایل موجود از نوع text را به عنوان ورودی و خروجی باز می کند
r+b	فایل موجود از نوع باینری را به عنوان ورودی و خروجی باز می کند
w+b	فایل موجود از نوع باینری را به عنوان ورودی و خروجی باز می کند
a+b	فایل موجود از نوع باینری را به عنوان ورودی و خروجی باز می کند (این فایل از قبل می تواند وجود داشته باشد).

به مثال ساده زیر توجه کنید:

```
FILE *fp;  // تعریف یک اشاره گر از نوع فایل
fp=fopen("c:/data.txt","rt")
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

در مثال بالا فایلی به نام data و با پسوند متنی txt بر روی درایو C و از نوع فقط خواندنی ایجاد می شود.

اگر فایل مورد نظر در شاخه موجود نباشد اشاره گر NULL برگردانده می شود که می توان به صورت زیر ایـن عمـل را چک نمود:

```
if ((fp=fopen("A:data.txt","w"))==NULL
 {
   printf("can not open file \n");
 }
```

بستن فایل:

بعد از اینکه برنامه نویس کارش را با فایل تمام کرد باید آن را ببندد. بستن تابع با دستور fclose() انجام می شود. کـه دارای الگوی زیر است:

int fclose(FILE *fp)

در الگوی بالا، fp به فایلی اشاره می کند که باید توسط تابع fclose() بسته شود. به عنـوان مثـال دسـتور fclose(p) فایلی را که p به آن اشاره می کند می بندد.

اگر چندین فایل بطور همزمان در برنامه باز باشند می توان آنها را با تابع fcloseall() بست. این تابع به صورت زیـر بـه کار می رود:

fcloseall();

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

در زیر به برخی از توابعی که در کار با فایل ها مورد استفاده قرار می گیرد آورده شده است:

نام تابع	توضیح

Putc() یا fputc()	از این دو تابع برای نوشتن کاراکتر در فایل استفاده می شود.
	الگوی تابع:
	int putc(int ch, FILE *fp)
	مثال:
	ch=getchar();
	putc(ch,in);

getc() یا fgetc()	برای خواندن کاراکتر ها از فایل استفاده دارد **الگوی تابع:** int getc(FILE *fp) **مثال:** FILE *in; ch=getc(in);
fputs()	برای نوشتن رشته ها در فایل بکار می رود. **الگوی تابع:** int puts(const char *str, FILE *fp)
fgets()	برای خواندن رشته از فایل کاربرد دارد. **الگوی تابع:** char *fgets(char *str,int length,FILE *fp)

rewind()	بردن موقعیت سنج به ابتدای فایل بدون اینکه بخواهیم فایل را بسته و دوباره آن را باز کنیم.الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: **void rewind(FILE *fp)**
ferror()	در حین انجام کار با فایل ممکن است خطایی رخ دهد. به عنوان مثال عدم وجود فضای کافی برای ایجاد فایل، آماده نبودن دستگاهی که فایل باید در آنجا تشکیل گردد و یا مواردی از این قبیل، منجر به بروز خطا می شوند. با استفاده از این تابع می توان از بروز چنین خطایی مطلع شوید. الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: **int ferror(FILE *fp)**
remove()	برای حذف فایلهای غیر ضروری می توان از این تابع استفاده کرد. الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: **int remove(char *filename)** در الگوی فوق filename به نام فایلی اشاره می کند که باید حذف شود. اگر عمل تابع با موفقیت انجام شود مقدار صفر وگرنه مقداری غیر از صفر برگردانده خواهد شد.
fflush()	انتقال کلیه داده های موجود در بافرها به فایل الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: **int fflush(FILE *fp)** اگر عمل تابع با موفقیت انجام شود مقدار صفر وگرنه مقدار EOF برگردانده خواهد شد.

fprintf()	نوشتن داده ها با فرمت خاصی بر روی فایل الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: int fprintf(FILE *fp,"*control_string,…",char arg,…) مثال: fprintf(fp,"%c",ch);
fscanf()	خواندن داده با فرمت خاص از فایل مورد نظر الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: int fscanf(FILE *fp,"control_string,…",char arg,…) **مثال:** fscanf(fp,"%d",&num);
fwrite()	مانند fprintf() می باشد اما با این تفاوت که دارای سرعت بیشتری نسبت به fprintf() می باشد. الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: int fwrite(void *buffer, int num_byte,int count,FILE *fp) مثال: Char table[20]; fwrite(table,sizeof(char),200,fp);
fread()	مانند fscanf() می باشد اما با این تفاوت که دارای سرعت بیشتری نسبت به آن می باشد. الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر است: int fread(void *buffer, int num_byte,int count,FILE *fp)

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

مثال: Char arr[10]; fread(arr,sizeof(char),10,fp);	

دسترسی تصادفی به فایل (ورودی-خروجی تصادفی)

مفهوم دستیابی تصادفی به فایل این است که در هرجایی از فایل بتوانیم بنویسیم و از هر جایی از فایل بتوانیم اطلاعات را بخوانیم. اگر بتوانیم " موقعیت سنج فایل" را به هر جایی که خواستیم، منتقل کنیم، به مفهوم تصادفی فایل دست می یابیم. برای تغییر مکان موقعیت سنج فایل از تابع fseek() استفاده می شود. الگوی این تابع در فایل stdio.h قرار داشته و به صورت زیر می باشد:

int fseek (FILE *fp,long num_byte,int origin)

اساس کار این تابع به این صورت است که با شروع از یک محل در فایل، که توسط پارامتر origin مشخص می شود، موقعیت سنج فایل را به طول num_byte تغییر مکان می دهد. یعنی تغییر مکان موقعیت سنج فایل به طور نسبی بوده، نسبت به محلی که توسط origin تعیین می گردد سنجیده می شود. به عنوان مثال اگر origin محل ۱۰۰ فایل را به عنوان مبدا حرکت مشخص کند و طول num_byte برابر ۲۰۰ باشد، موقعیت سنج فایل به محل ۳۰۰ منتقل خواهد شد. مقادیری که origin می تواند بپذیرد در جدول زیر آورده شده است:

مبدا حرکت	نام ماکرویی که بیانگر این مبدا است
ابتدای فایل	SEEK_SET
موقعیت فعلی فایل	SEEK_CUR
انتهای فایل	SEEK_END

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

توابع زیر از MSDN VC++ آورده شده است:

Visual C++

تابع ftell() :

کار این تابع موقعیت فعلی اشاره گر فایل یا همان File Pointer را بر می گرداند. الگوی این تابع در فایل stdio.h قرار دارد و به صورت زیر می باشد:

long ftell(FILE *stream);

مثال:

این برنامه یک فایل با نام FTELL.C باز می کند و 100 کاراکتر از آن فایل را می خواند. تابع ftell() موقعیت اشاره گر فایل را شناسایی کرده و موقعیت آنرا به ما می دهد که در نهایت عدد 100 به عنوان خروجی به ما داده می شود.

```
#include <stdio.h>

FILE *stream;

void main( void )
{
   long position;
   char list[100];
   if( (stream = fopen( "ftell.c", "rb" )) != NULL )
   {
      /* Move the pointer by reading data: */
      fread( list, sizeof( char ), 100, stream );
      /* Get position after read: */
      position = ftell( stream );
      printf( "Position after trying to read 100 bytes: %ld\n",position );
      fclose( stream );
   }
}
```

Output

```
Position after trying to read 100 bytes: 100
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

تابع _open() ، _wopen()

کار دو تابع فوق باز کردن فایل می باشد.

الگوی آنها به صورت زیر می باشد:

```
int _open( const char *filename, int oflag [, int pmode] );
int _wopen( const wchar_t *filename, int oflag [, int pmode] );
```

تابع _open() در io.h ، و تابع _wopen() در io.h و یا wchar.h می باشد.

مقادیر برگشتی این توابع:

این تابع ها عدد ۱- را در صورت بروز خطا بر می گرداند که case خطا دارای مقادیر زیر بسته به نوع خطا می باشد:

مقدار برگشتی	توضیح
EACCES	سعی بر نوشتن بر روی فایل فقط خواندنی یا فایلهای به اشتراک گذاشته ای که کاربر اجازه تغییر در آنها را نداشته باشد و یا اینکه مسیر داده شده یک پوشه یا دایرکتوری باشد.
EEXIST	این فایل با این نام قبلا وجود دارد.
EINVAL	اشتباه بودن آرگومانهای oflag یا pmode
EMFILE	تعداد فایل های باز بسیار بسیار زیاد است.
ENOENT	پیدا نشدن فایل یا مسیر مورد نظر

آرگومان oflag() یک عبارت عددی است که دارای مقادیر ثابت طبق جدول زیر می باشد:

ترکیبات جدول زیر در FCNTL.H قرار دارد.

مقدار برگشتی	توضیح
_O_APPEND	قبل از هرگونه عملیات نوشتنی بر روی فایل اشاره گر فایل را به انتهای آن می برد.
_O_BINARY	فایل را در حالت باینری باز می کند.
_O_CREAT	ایجاد کردن و باز کردن یک فایل جدید برای نوشتن. اگر فایل قبلا موجود باشد تفاوتی نمی کند. در این حالت ویژه آرگومان pmode را برابر _O_CREAT قرار می دهیم.
_O_CREAT | _O_SHORT_LIVED	در صورتی که نتوانیم قسمتی از حافظه را پاک کنیم برای ما یک فایل موقت ایجاد می کند. در این حالت خاص آرگومان pmode را برابر _O_CREAT قرار می دهیم.
_O_CREAT | _O_TEMPORARY	هنگامی که آخرین فایل بسته شد فایل حذف می گردد. در این حالت خاص آرگومان pmode را برابر _O_CREAT قرار می دهیم.
_O_CREAT | _O_EXCL	اگر نام فایل وجود داشته باشد مقدار خطا را بر می گرداند و باید با _O_CREAT همراه باشد.
_O_RANDOM	دسترسی اولیه تصادفی به دیسک
_O_RDONLY	فایل را به صورت فقط خواندنی باز می کند این آرگومان با فلگ های _O_RDONLY و _O_WRONLY استفاده نمی شود.

_O_RDWR	باز کردن فایل به صورت خواندنی- نوشتنی این آرگومان با فلگ های _O_RDONLY و _O_WRONLY استفاده نمی شود.
_O_SEQUENTIAL	دسترسی اولیه ترتیبی به دیسک
_O_TEXT	باز کردن فایل متنی
_O_TRUNC	حذف کردن اطلاعات در فایل تا اندازه آن فایل به صفر برسد. فایل مورد نظر باید اجازه نوشتن بر روی آن داده شده باشد و محافظت شده نباشد. فلگی که نمی توان استفاده کرد: _O_RDONLY با _O_TRUNC برای باز کردن یک _O_CREAT فایل موجود و یا ایجاد یک فایل جدید به کار می رود.
_O_WRONLY	باز کردن فایل به صورت فقط نوشتنی. با دو فلگ زیر استفاده نمی شود: _O_RDWR یا _O_WRONLY در ضمن یک مقدار مشخصی برای دسترسی وجود ندارد.

آرگومان pmode() فقط هنگامی به کار می رود که _O_CREAT آمده باشد. اگر فایل از قبل موجود بود، pmode چشم پوشی می شود در غیر اینصورت pmode ، اجازه تنظیمات مختلف فایل را به ما می هد.

S_IREAD: مجوز فقط خواندن

_S_IWRITE : مجوز نوشتن (اجازه موثر و کارا برای خواندن و نوشتن)

_S_IREAD | _S_IWRITE : مجوز نوشتن و خواندن

برای روشن تر شدن مطلب به مثال زیر توجه کنید:

این مثال به توضیح کاربرد دستور _open می پردازد. یک فایل با نام OPEN.C برای ورودی و یک فایل با نام OPEN.OUT برای خروجی در نظر گرفته شده است. سپس فایلها بسته می شوند.

Example

```
/* OPEN.C: This program uses _open to open a file
 * named OPEN.C for input and a file named OPEN.OUT
 * for output. The files are then closed.
 */

#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>

void main( void )
{
   int fh1, fh2;

   fh1 = _open( "OPEN.C", _O_RDONLY );
   if( fh1 == -1 )
      perror( "open failed on input file" );
   else
   {
      printf( "open succeeded on input file\n" );
      _close( fh1 );
   }

   fh2 = _open( "OPEN.OUT", _O_WRONLY | _O_CREAT, _S_IREAD | _S_IWRITE );
   if( fh2 == -1 )
      perror( "Open failed on output file" );
   else
   {
      printf( "Open succeeded on output file\n" );
      _close( fh2 );
   }
}
```

Output

```
Open succeeded on input file
Open succeeded on output file
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

تابع _close()

از این تابع برای بستن فایل استفاده می شود. الگوی این تابع به صورت زیر است و در io.h قرار دارد:

int _close(int *handle* **);**

بعد از اجرای این این تابع اگر مقدار صفر برگردانده شود به مفهوم این می باشد که فایل بسته شده در غیر اینصورت errno به EBADF تنظیم می شود.

به مثال زیر توجه نمایید:

این مثال به توضیح کاربرد دستور _close می پردازد. یک فایل با نام OPEN.C برای ورودی و یک فایل با نام OPEN.OUT برای خروجی در نظر گرفته شده است. سپس فایلها بسته می شوند.

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>

void main( void )
{
   int fh1, fh2;

   fh1 = _open( "OPEN.C", _O_RDONLY );
   if( fh1 == -1 )
      perror( "open failed on input file" );
   else
   {
      printf( "open succeeded on input file\n" );
      _close( fh1 );
   }

   fh2 = _open( "OPEN.OUT", _O_WRONLY | _O_CREAT, _S_IREAD |
                            _S_IWRITE );
   if( fh2 == -1 )
      perror( "Open failed on output file" );
   else
   {
      printf( "Open succeeded on output file\n" );
      _close( fh2 );
   }
}
```

Output

```
Open succeeded on input file
Open succeeded on output file
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

توابع _lseek, _lseeki64

حرکت دادن موقعیت سنج فایل به یک محل خاص (دلخواه)

الگوی این توابع به صورت زیر می باشد و در io.h قرار دارند:

long _lseek(int *handle*, **long** *offset*, **int** *origin* **);**

__int64 _lseeki64(int *handle*, **__int64** *offset*, **int** *origin* **);**

مقدار برگشتی _lseek() در قالب long از موقعیت جدید از ابتدای فایل می باشد.

مقدار برگشتی _lseeki64 یک عدد ۶۴ بیتی می باشد.

در صورت بروز خطا تابع مقدار 1L- را بر می گرداند.

توضیحات مربوط به این دستور قبلا توضیح داده شده است که در این قسمت از یادآوری مجدد آن پرهیز می کنیم.

در جدول زیر مقادیری را که orogin می تواند بگیرد آورده شده است:

مبدا حرکت	نام ماکرویی که بیانگر این مبدا است
ابتدای فایل	SEEK_SET
موقعیت فعلی فایل	SEEK_CUR
انتهای فایل	SEEK_END

به مثال زیر توجه نمایید.

Example

```
/* LSEEK.C: This program first opens a file named LSEEK.C.
 * It then uses _lseek to find the beginning of the file,
 * to find the current position in the file, and to find
 * the end of the file.
 */

#include <io.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>

void main( void )
{
   int fh;
   long pos;               /* Position of file pointer */
   char buffer[10];

   fh = _open( "lseek.c", _O_RDONLY );

   /* Seek the beginning of the file: */
   pos = _lseek( fh, 0L, SEEK_SET );
   if( pos == -1L )
      perror( "_lseek to beginning failed" );
   else
      printf( "Position for beginning of file seek = %ld\n", pos );

   /* Move file pointer a little */
    _read( fh, buffer, 10 );

   /* Find current position: */
   pos = _lseek( fh, 0L, SEEK_CUR );
   if( pos == -1L )
      perror( "_lseek to current position failed" );
   else
      printf( "Position for current position seek = %ld\n", pos );

   /* Set the end of the file: */
   pos = _lseek( fh, 0L, SEEK_END );
   if( pos == -1L )
      perror( "_lseek to end failed" );
   else
      printf( "Position for end of file seek = %ld\n", pos );

   _close( fh );
}
```

Output

```
Position for beginning of file seek = 0
Position for current position seek = 10
Position for end of file seek = 1207
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

تابع _read()

از این تابع برای خواندن اطلاعات از یک فایل استفاده می شود.

الگوی آن به صورت زیر است و در io.h قرار دارد.

int _read(int *handle*, **void *** *buffer*, **unsigned int** *count* **);**

این تابع تعداد بایت های خوانده شده از فایل را بر می گرداند. اگر مقدار برگشتی کمتر و یا مساوی صفر باشد خطا رخ می دهد.

برای درک بهتر این تابع به مثال زیر توجه کنید:

```
/* READ.C: This program opens a file named
 * READ.C and tries to read 60,000 bytes from
 * that file using _read. It then displays the
 * actual number of bytes read from READ.C.
 */
#include <fcntl.h>      /* Needed only for _O_RDWR definition */
#include <io.h>
#include <stdlib.h>
#include <stdio.h>

char buffer[60000];

void main( void )
{
   int fh;
   unsigned int nbytes = 60000, bytesread;

   /* Open file for input: */
   if( (fh = _open( "read.c", _O_RDONLY )) == -1 )
   {
      perror( "open failed on input file" );
      exit( 1 );
   }

   /* Read in input: */
   if( ( bytesread = _read( fh, buffer, nbytes ) ) <= 0 )
      perror( "Problem reading file" );
   else
      printf( "Read %u bytes from file\n", bytesread );

   _close( fh );
}
```

Output

```
Read 775 bytes from file
```

محقق: محمد بشیری
آدرس پست الکترونیک: m.bashiry@gmail.com
آدرس سایت: http://bashiry.250free.com

تابع _write()

با استفاده از این تابع می توان اطلاعات را در فایل نوشت.

الگوی این تابع به شکل زیر می باشد و در io.h قرار دارد.

int _write(int *handle*, **const void ***buffer*, **unsigned int** *count* **);**

اگر عمل نوشتن با موفقیت انجام شود مقدار برگشتی تابع تعداد بایت های نوشته شده خواهد بود.

مثال زیر یک فایل را به عنوان خروجی باز می کند و با استفاده از این تابع بایت هایی را در آن می نویسد.

Example

```
/* WRITE.C: This program opens a file for output
 * and uses _write to write some bytes to the file.
 */

#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

char buffer[] = "This is a test of '_write' function";

void main( void )
{
   int fh;
   unsigned byteswritten;

   if( (fh = _open( "write.o", _O_RDWR | _O_CREAT,
                              _S_IREAD | _S_IWRITE )) != -1 )
   {
      if(( byteswritten = _write( fh, buffer, sizeof( buffer ))) == -1 )
         perror( "Write failed" );
      else
         printf( "Wrote %u bytes to file\n", byteswritten );

      _close( fh );
   }
}
```

Output

```
Wrote 36 bytes to file
```